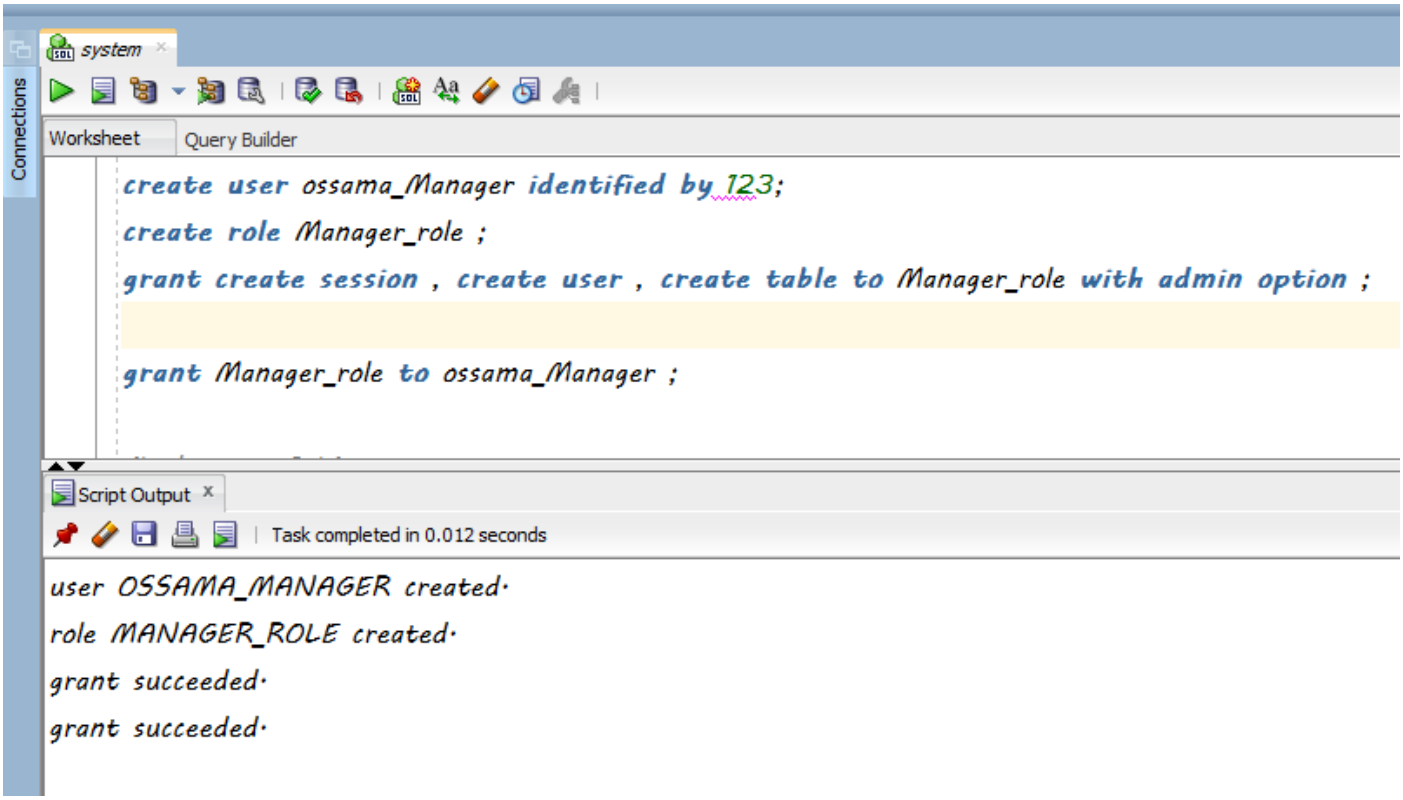
ORACLE TASK:

An organization has a "Department" table and an "Employees" table in Oracle. The "Department" table contains information about different departments in the organization, and the "Employees" table contains information about the employees, including the department they belong to. The department table contains ID as a primary key and department name. The departments are HR, IT, and finance. The Employee table contains ID as a primary key and name, salary, and department ID as a foreign key.

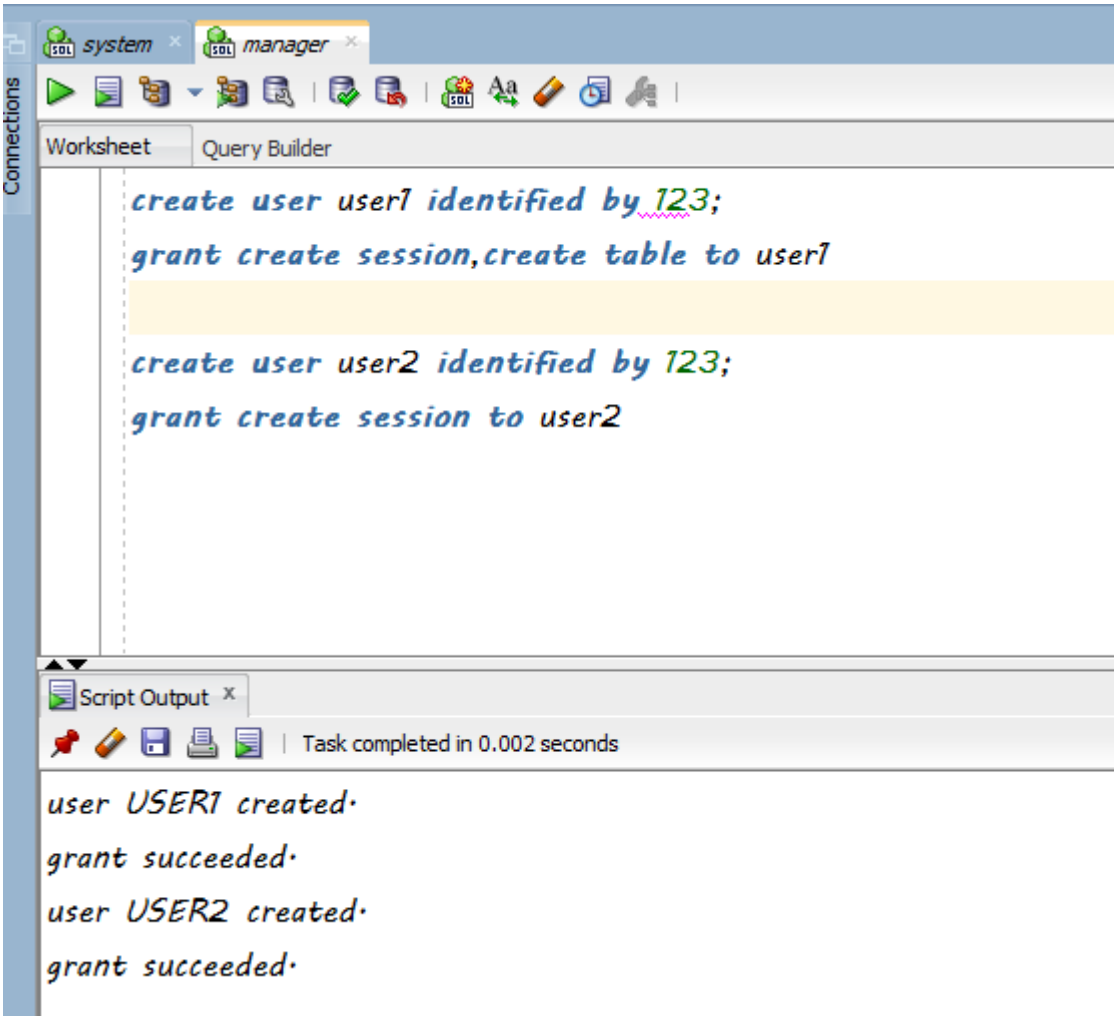
a) Create a Manager User and grant them a role of privileges to create two users. Let User 1 create the Employee and the Department table. Let User 2 insert 5 rows of employees.

```
system create manager and grant privileges to create session and
create user and create table with admin option to grant it to user1
```



```
/* alter user1 */
alter user user1 quota 100M on system;
```

Alter user1 to can create tables



Manager create two users and grant create session to them

And grant create table to only user1

system | manager | user1_conn

Worksheet | Query Builder

```
create table Department (
  id number primary key,
  department_name varchar2(50) not null);
create table Employees (
  id number primary key,
  name varchar2(100) not null,
  salary number not null,
  department_id number references Department(id));

insert into Department values(1,'HR');
insert into Department values(2,'IT');
insert into Department values(3,'finance');

grant insert on Employees to user2;
```

Script Output | Task completed in 0.009 seconds

```
table DEPARTMENT created.
table EMPLOYEES created.
1 rows inserted.
1 rows inserted.
1 rows inserted.
grant succeeded.
```

User1 create two table department and employees and insert there rows in table department

```
insert into user1.Employees values (1, 'John Doe', 500.00, 1);
insert into user1.Employees values (2, 'Jane Smith', 600.00, 2);
insert into user1.Employees values (3, 'Bob Johnson', 550.00, 1);
insert into user1.Employees values (4, 'Alice Brown', 700.00, 2);
insert into user1.Employees values (5, 'Charlie Davis', 650.00, 2);

commit;
```

```
Task completed in 0.001 seconds
1 rows inserted.
1 rows inserted.
1 rows inserted.
1 rows inserted.
1 rows inserted.
committed.
```

User2 insert 5 rows in table department and commit

```
/* select from employees table*/
select * from employees;
```

All Rows Fetched: 5 in 0.021 seconds

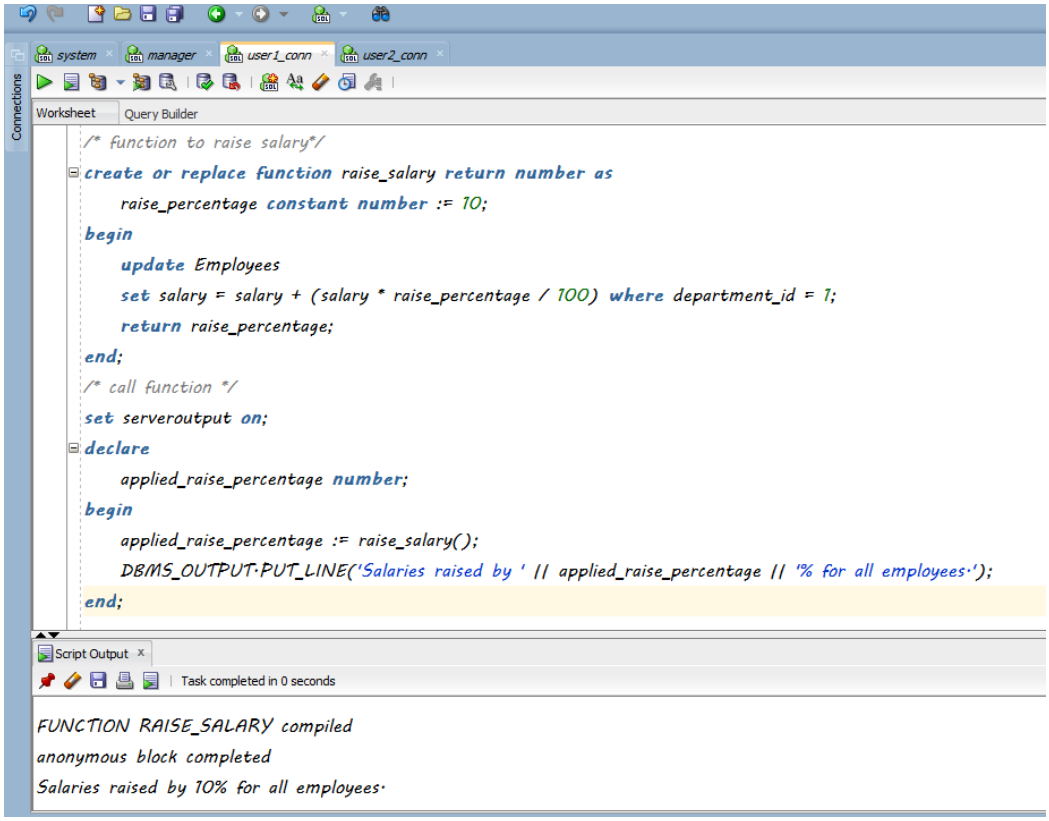
	ID	NAME	SALARY	DEPARTMENT_ID
1	1	John Doe	500	1
2	2	Jane Smith	600	2
3	3	Bob Johnson	550	1
4	4	Alice Brown	700	2
5	5	Charlie Davis	650	2

User1 select all rows in employees after user2 commit

b) Demonstrate generating a blocker-waiting situation using two transactions by user 1 and user 2. The Transaction is calling a function that raises the rate of salary by 10% for department 1.

```
/* grant create function to user2*/
grant create procedure to  user1;
```

System grant user1 to crate function or procedure

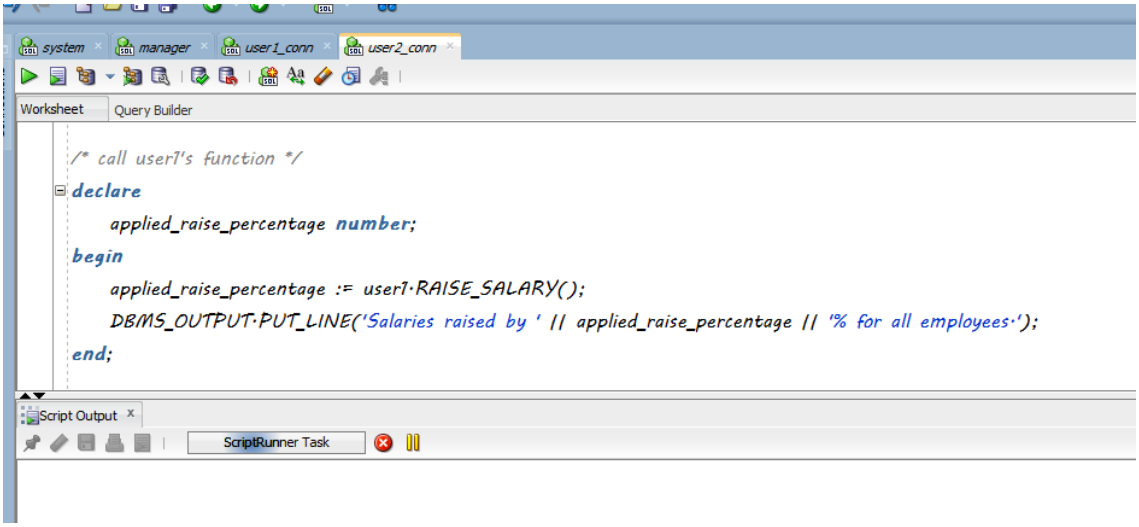


User1 create function to raise salary for department 1 by 10% and execute function

```
/* grant execute function to user2*/
grant execute on raise_salary to user2;
```

User1 grant execute function raise salary to user2

c) Identify the sessions in the situation using SID and serial# for both blocker and waiting sessions.



User2 execute raise_salary and Enters a waiting state

```
/* select SID and sreail# for blocker and waiting usres*/
select
    blocking_session.sid as blocking_sid,
    blocking_session.SERIAL# as blocking_serial,
    waiting_session.sid as waiting_sid,
    waiting_session.SERIAL# as waiting_serial
from
    v$session blocking_session,
    v$session waiting_session
where
    blocking_session.sid = waiting_session.BLOCKING_SESSION;
```

All Rows Fetched: 1 in 0.015 seconds

	BLOCKING_SID	BLOCKING_SERIAL	WAITING_SID	WAITING_SERIAL
1	113	453	135	141

System select blocker an waiting info

d) Demonstrate a deadlock scenario and display the expected result.

```
/* update employee 1 */
update Employees set salary = salary + salary * 0.1 where Employees.ID = 1;
```

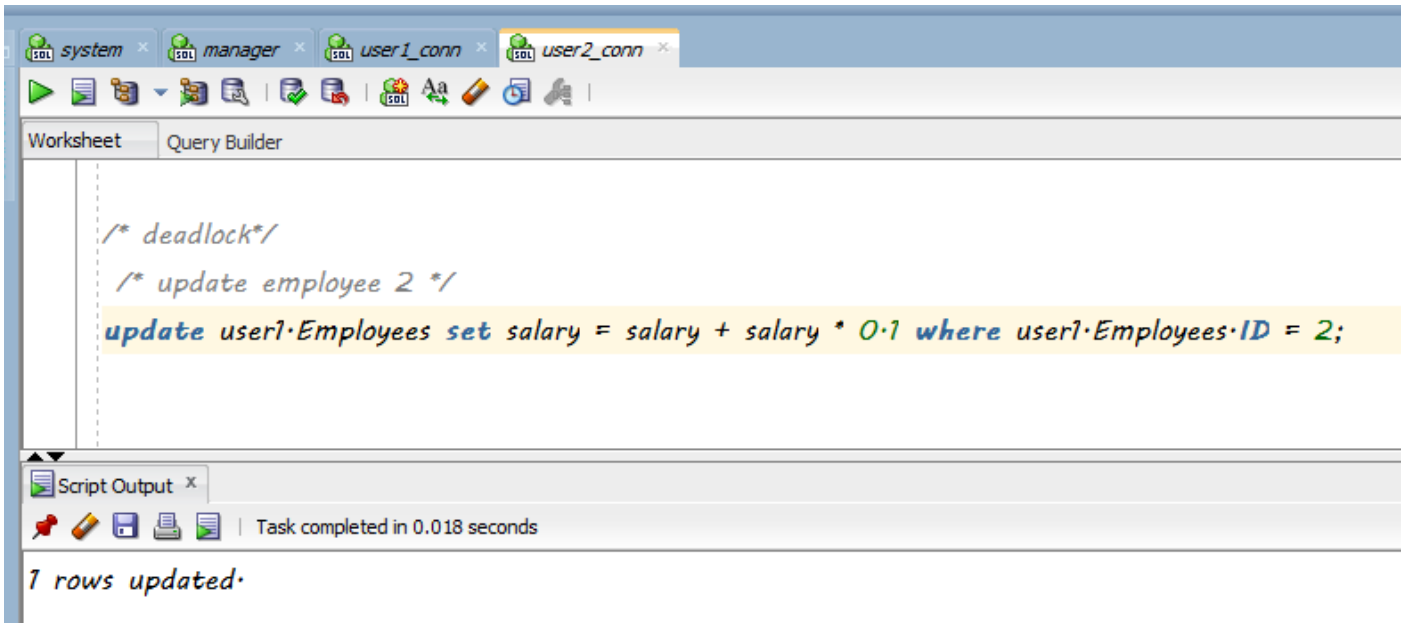
Task completed in 0.002 seconds

```
1 rows updated.
```

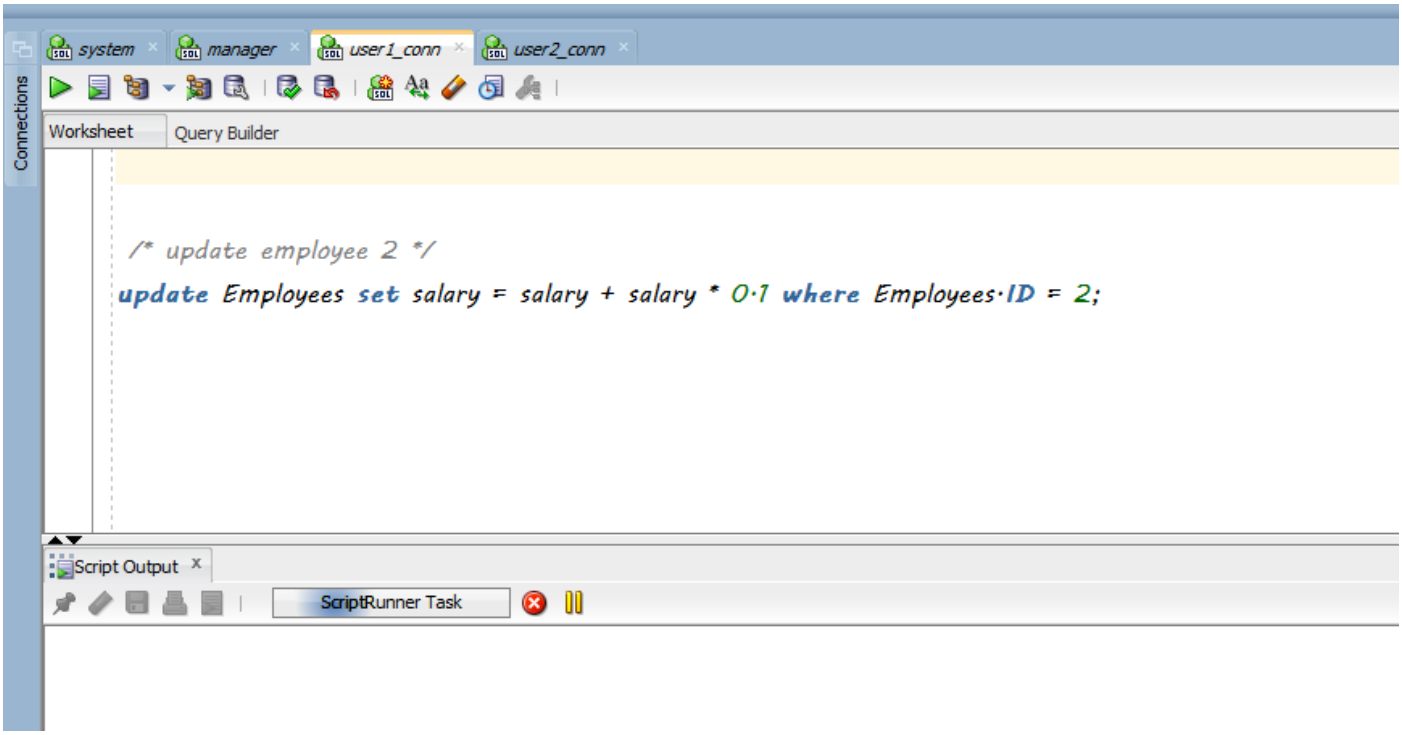
User1 update employee have id = 1 in table employees
But don't make commit

```
/* deadlock*/
grant update on Employees  to user2;
```

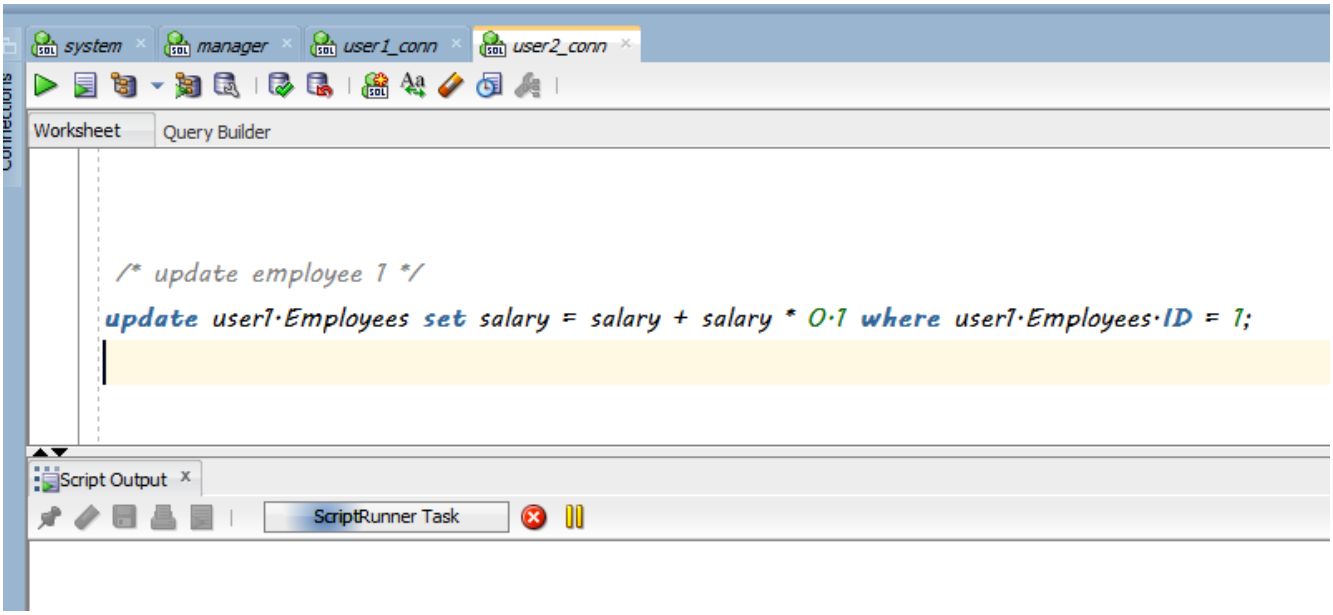
User1 grant update on employees table employees
But granting done before user1 make update



User2 update employee have id = 2 in table employees
But don't make commit



user1 try to update employee have id = 2 and Enters a waiting state



user2 try to update employee have id = 1 and Enters a waiting state

system | manager | user1_conn | user2_conn

Worksheet | Query Builder

```
update Employees set salary = salary + salary * 0.1 where Employees.ID = 1;


/* update employee 2 */
update Employees set salary = salary + salary * 0.1 where Employees.ID = 2;
```

Script Output

Task completed in 39.189 seconds

```
Error starting at line : 54 in command -
update Employees set salary = salary + salary * 0.1 where Employees.ID = 2
Error report -
SQL Error: ORA-00060: deadlock detected while waiting for resource
00060. 00000 -  "deadlock detected while waiting for resource"
*Cause:    Transactions deadlocked one another while waiting for resources.
*Action:   Look at the trace file to see the transactions and resources
           involved. Retry if necessary.
```

Each user in waiting state that is called deadlock

e) Perform the following functions

i. Create a function that calculates the average salary for any department

```
/*Create a function that calculates the average salary for any department */
create or replace function avg_salary(dept_id number)
return number IS avg_sal number;
begin
    select avg(salary)
    into avg_sal
    from employees
    where department_id = dept_id;
    return avg_sal;
end;
/* executer avg_salary function */
set serveroutput on;
declare avg_sal number;
begin
    avg_sal := avg_salary(1);
    DBMS_OUTPUT.PUT_LINE('Average Salary is: ' || avg_sal);
end;
```

Script Output

Task completed in 0.002 seconds

```
anonymous block completed
Average Salary is: 525
```

ii. Create a function that calculates the Total Salary in a Department.

```
/*Create a function that calculates the Total Salary in a Department· */
create or replace function  total_salary(dept_id number)
return number is total_sal number;
begin
    select sum(salary)
    into total_sal
    from employees
    where department_id = dept_id;
    return total_sal;
end;

/* execute total salary function*/
declare total_sal number;
begin
    total_sal := total_salary(2);
    DBMS_OUTPUT·PUT_LINE('Total Salary is :'|| total_sal);
end;
```

```
anonymous block completed
Total Salary is :1950
```

iii. Create a function that calculates the maximum Salary.

system | manager | user1_conn | user2_conn | EMPLOYEES

Worksheet | Query Builder

```
/*Create a function that calculates the maximum Salary*/
create or replace function max_salary return number
is max_sal number;
begin
    select max(salary)
    into max_sal
    from employees;
    return max_sal;
end;

/* execute max_salary function */
declare
    max_sal number;
begin
    max_sal := max_salary();
    DBMS_OUTPUT.PUT_LINE('Maximum Salary: ' || max_sal);
end;
```

Script Output

Task completed in 0.003 seconds

```
anonymous block completed
Maximum Salary: 700
```

Team Members :

Students of the Faculty of Computers and Artificial Intelligence at Helwan University, Department of Computer Science

Course : Database-II(fall 2023)

Name	ID
Ossama Samir	**20210140**
Abd-ulla Mohamed	**20210556**
Marowa Omar	**20210900**
Rana Essam	**20210335**
Toka Mohamed	**20210242**
Abd-ulla Yasser	**20210564**
Omar Nasser	**20210628**
Seif El-den Samy	**20210437**